ИНФОРМАТИКА

2006 **январь-март** **№ 1**

УДК 519.68, 519.7

А.Д. Закревский, Н.Р. Торопов

РЕШЕНИЕ БОЛЬШИНСТВА ЛИНЕЙНЫХ ЛОГИЧЕСКИХ УРАВНЕНИЙ НЕСОВМЕСТНОЙ СИСТЕМЫ НА СУПЕРКОМПЬЮТЕРЕ

Рассматриваются переопределенные системы линейных логических уравнений, число уравнений в которых превышает число неизвестных. Такие системы, как правило, не имеют корней и называются поэтому несовместными. Однако и они могут быть разрешимы в определенном смысле. Например, для криптографии представляет интерес задача поиска корней, удовлетворяющих максимальному числу уравнений, или, в случае искажения правых частей уравнений, задача восстановления системы. Предлагается параллельная реализация рандомизированного алгоритма решения несовместных систем на суперкомпьютере семейства СКИФ. Приводятся результаты экспериментальных испытаний алгоритма, показывающие эффективность параллельных вычислений при решении больших систем линейных логических уравнений.

Введение

Использование суперкомпьютеров для решения трудных задач эффективно, если вычислительный процесс декомпозируется, распараллеливается и может быть поэтому реализован на множестве одновременно работающих процессоров. Это условие обеспечивается разработкой параллельных алгоритмов решения рассматриваемых задач, иллюстрируемой в данной статье на примере решения несовместных систем линейных логических уравнений (СЛЛУ).

Рассмотрим СЛЛУ, содержащую m уравнений с n неизвестными:

$$\begin{aligned} a_1^{\,1}x_1 \oplus a_1^{\,2}x_2 \oplus \ldots \oplus a_1^{\,n}x_n &= y_1 ; \\ a_2^{\,1}x_1 \oplus a_2^{\,2}x_2 \oplus \ldots \oplus a_2^{\,n}x_n &= y_2 ; \\ \ldots & \\ a_m^{\,1}x_1 \oplus a_m^{\,2}x_2 \oplus \ldots \oplus a_m^{\,n}x_n &= y_m . \end{aligned}$$

Коэффициенты a_i^j, свободные члены y_i и переменные x_j принимают значения из множества $\{0, 1\}$, а символ $\oplus$ представляет оператор сложения по модулю два. В компактной векторно-матричной форме эта система описывается уравнением

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y},$$

где $\boldsymbol{A}$ – булева $m \times n$ матрица коэффициентов; $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ – булев вектор переменных и $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ – булев вектор свободных членов. Роль внутренней операции векторно-матричного произведения играет конъюнкция, а роль внешней – дизъюнкция с исключением (сумма по модулю два).

Решить СЛЛУ значит найти ее корень – набор значений переменных $\boldsymbol{x}$, удовлетворяющий всем уравнениям, т. е. обращающий их в тождества. Система бывает *определенной* (имеющей единственный корень), *неопределенной* (обладающей двумя или более корнями) или *переопределенной* (называемой также *несовместной*, т. е. не имеющей корней) [1].

Характерным для линейных уравнений является то, что они разрешимы относительно любой переменной. На использовании этого свойства базируется метод исключения переменных, предложенный Гауссом для линейных алгебраических уравнений с действительными переменными [2] и легко модифицируемый применительно к логическим уравнениям [3]. Данный метод дает ответ на вопрос, совместна ли система, и находит единственный корень определенной системы либо множество всех корней неопределенной системы. Полученный при этом результат является удовлетворительным для определенной системы, однако он недостаточен для неопределенной и тем более для несовместной систем.

При рассмотрении ряда практических проблем возникают задачи нахождения оптимального решения неопределенной СЛЛУ. Например, при синтезе логических AND/EXOR-схем приходится искать кратчайший корень неопределенной системы, представляемый n-мерным булевым вектором $\boldsymbol{x}$ с минимальным числом единиц [3]. В случае несовместной системы можно поставить задачу удовлетворения максимального числа уравнений или восстановления системы с искаженными свободными членами [4]. Эти задачи представляют интерес с точки зрения криптографии.

1. Решение неопределенных СЛЛУ

Решение неопределенной СЛЛУ $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$, имеющей при $m < n$ в общем случае 2^{n-m} корней, чаще всего сводится к поиску одного из таких корней, удовлетворяющего определенному заданному критерию. Например, отыскивается *кратчайший корень*, содержащий в представляющем его векторе $\boldsymbol{x}$ минимальное число единичных компонент.

Поиск кратчайшего решения неопределенной СЛЛУ базируется на *методе Гаусса*, суть которого заключается в канонизации *расширенной* матрицы $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{y})$, получаемой из основной матрицы коэффициентов $\boldsymbol{A}$ путем столбцовой конкатенации ее с вектор-столбцом $\boldsymbol{y}$ [5]. Пусть ранг $r(\boldsymbol{S})$ матрицы $\boldsymbol{S}$ (максимальное число линейно независимых столбцов в ней) равен m. В результате канонизации матрица $\boldsymbol{S}$ преобразуется в *базис* – квадратный столбцовый минор с одной единицей в каждом ряду. Остальные $n - m$ столбцов матрицы $\boldsymbol{A}$ в преобразованном виде составляют *остаток*. Каждое из 2^{n-m} подмножеств столбцов из остатка однозначно определяет соответствующее решение рассматриваемой системы, в которое кроме данного подмножества включаются также некоторые столбцы базиса, что обеспечивает равенство покомпонентной суммы по модулю два всех включенных в решение столбцов столбцу $\boldsymbol{y}$ (также в преобразованном виде). Число w столбцов, вошедших в решение, называется его *весом*.

При поиске кратчайшего решения на базе канонической формы производится перебор подмножеств столбцов в остатке, рассмотрение определяемых ими решений и выбор кратчайшего из них (с минимальным весом). Если при этом априори известно, что вес w кратчайшего решения не превышает некоторого значения γ, то уровень перебора (мощность перебираемых подмножеств) ограничивается этой величиной. Если $\gamma \ge n - m$, то приходится перебирать все 2^{n-m} подмножеств.

Математическое ожидание $\gamma(m, n)$ веса кратчайшего решения случайной неопределенной системы уравнений с параметрами m и n оценивается в работе [6] максимальным значением параметра k, при котором $\beta(m, n, k) < 1$, т. е. $\gamma(m, n) = k$ при $\beta(m, n, k) < 1 \le \beta(m, n, k+1)$, где через $\beta(m, n, k)$ обозначено математическое ожидание числа решений, вес которых не превышает k:

$$\beta(m, n, k) = \sum_{i=0}^{k} C_n^i \, 2^{-m}.$$

Очевидно, что объем вычислений при поиске кратчайшего решения пропорционален числу $N(m, n)$ рассматриваемых подмножеств столбцов остатка, которое в сильной степени зависит от уровня перебора, равного $k = \gamma(m, n)$. В данном методе

$$N(m, n) = \sum_{i=0}^{k} C_{n-m}^{\,i}.$$

Уровень перебора значительно снижается в декомпозиционных методах решения неопределенных СЛЛУ, что на много порядков ускоряет решение данной задачи [1, 6].

2. Решение несовместных СЛЛУ

Несовместные (или переопределенные) системы, хотя и не имеют корней, могут быть разрешимы в определенном смысле.

Будем рассматривать несовместную систему $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$ с заданными $\boldsymbol{A}$ и $\boldsymbol{y}$. Положим, что $m > n$, матрица $\boldsymbol{A}$ и вектор $\boldsymbol{y}$ случайны (их компоненты принимают одно из двух значений, 0 или 1, с вероятностью 0,5) и все столбцы матрицы $\boldsymbol{A}$ линейно независимы. Тогда среди 2^m различных значений булева m-вектора найдутся 2^n значений (соответствующих различным значениям n-вектора $\boldsymbol{x}$), подстановка которых вместо $\boldsymbol{y}$ делает систему совместной. Эти значения, рассеянные в m-мерном булевом пространстве $M(\boldsymbol{y})$, называются *пригодными*.

В работе [6] сформулированы три комбинаторно эквивалентные друг другу задачи для таких систем, представляющие интерес в криптографии и в логическом проектировании. Все они одинаково сложны с вычислительной точки зрения и фактически представляют различные интерпретации одной и той же комбинаторной задачи.

1. *Поиск ближайшего к вектору* $\boldsymbol{y}$ *пригодного значения* – изменение в векторе $\boldsymbol{y}$ значения минимального числа компонент с тем, чтобы полученное значение вектора оказалось пригодным, или, иными словами, нахождение корректирующего вектора $\boldsymbol{e}$, такого, что $\boldsymbol{y} \oplus \boldsymbol{e} = \boldsymbol{y}^*$, где $\boldsymbol{y}^*$ – ближний к $\boldsymbol{y}$ пригодный вектор.

2. *Решение максимального числа уравнений* – нахождение значения вектора $\boldsymbol{x}$, удовлетворяющего максимальному числу уравнений в системе (обращающего их в тождества).

3. *Линейная аппроксимация частичной булевой функции* – нахождение оптимальной аппроксимации линейной функцией $f(\boldsymbol{x})$, совпадающей по значениям с функцией $y(\boldsymbol{x})$ на максимальном числе строк, если рассматривать систему из матрицы $\boldsymbol{A}$ и вектора $\boldsymbol{y}$ как частичную булеву функцию $y(\boldsymbol{x})$, определенную на строках матрицы $\boldsymbol{A}$.

В работе [6] показано, что решение этих задач для переопределенной системы с m уравнениями и n переменными сводится к нахождению кратчайшего корня неопределенной системы с $m–n$ уравнениями и m переменными, получаемой из исходной системы средствами теории линейных векторных пространств.

2.1. Оценка возможности восстановления СЛЛУ с искаженной правой частью

В работе [4] сформулированы условия, при которых постановка задачи о восстановлении случайной несовместной системы $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ с искаженной правой частью корректна. Пусть $\boldsymbol{y} = \boldsymbol{y}^* \oplus \boldsymbol{e}$, где $\boldsymbol{y}^*$ – некоторый вектор, при котором система совместна, а $\boldsymbol{e}$ – так называемый *вектор искажений* с весом $w(\boldsymbol{e})$, равным числу компонент со значением 1. Таким образом, вектор $\boldsymbol{y}$, представляющий результат искажения пригодного вектора $\boldsymbol{y}^*$, оказывается на расстоянии $w(\boldsymbol{e})$ от вектора $\boldsymbol{y}^*$ (по Хэммингу). Задача заключается в нахождении вектора $\boldsymbol{y}^*$ по заданной матрице $\boldsymbol{A}$ и вектору $\boldsymbol{y}$. Очевидно, что вектор $\boldsymbol{y}^*$ можно отыскать в пространстве $M(\boldsymbol{y})$ на сфере радиусом $w(\boldsymbol{e})$ и с центром в точке $\boldsymbol{y}$. Однако при достаточно большом радиусе сферы он может затеряться среди множества попадающих в сферу других пригодных векторов, также обеспечивающих совместность рассматриваемой системы.

Математическое ожидание числа пригодных значений вектора $\boldsymbol{y}$, расположенных на сфере с радиусом k и с центром в случайной точке пространства $M(\boldsymbol{y})$, обозначаемое через $\rho(m, n, k)$, находится путем умножения числа элементов в рассматриваемой области на вероятность пригодности случайно выбранного элемента, т. е. $\rho(m, n, k) = C_m^k\, 2^{n-m}$.

Аналогично через $\sigma(m, n, k)$ обозначается математическое ожидание числа пригодных значений вектора $\boldsymbol{y}$, расположенных в шаре с радиусом k и с центром в случайной точке пространства $M(\boldsymbol{y})$:

$$\sigma(m, n, k) = \sum_{i=0}^{k} C_m^i\, 2^{n-m}.$$

Очевидно, что данную задачу можно решить лишь при таких значениях параметра $w(\boldsymbol{e})$, при которых значение функции $\sigma(m, n, w(\boldsymbol{e}))$ будет достаточно мало, например меньше единицы. В связи с этим рассматривается максимальное значение τ параметра $w(\boldsymbol{e})$, удовлетворяющее условию $\sigma(m, n, \tau) < 1 \le \sigma(m, n, \tau+1)$ и определяемое как некоторая функция $\tau(m, n)$. Таким образом, постановка задачи о восстановлении вектора $\boldsymbol{y}^*$ корректна, если $w(\boldsymbol{e}) \le \tau(m, n)$, а также если $w(\boldsymbol{e}) \ge m - \tau(m, n)$, поскольку $C_m^{\,k} = C_m^{\,m-k}$.

2.2. Прямой метод решения несовместной СЛЛУ

Матрица $\boldsymbol{A}$ порождает *линейное векторное пространство* $\mathrm{Sp}(\boldsymbol{A})$, образованное всеми различными покомпонентными суммами по модулю два всех возможных подмножеств столбцов из $\boldsymbol{A}$. Следовательно, система $\boldsymbol{Ax} = \boldsymbol{y}$ совместна, если и только если $\boldsymbol{y} \in \mathrm{Sp}(\boldsymbol{A})$. В этом случае вектор $\boldsymbol{y}$ является пригодным. Рассматриваемая задача решения несовместной системы $\boldsymbol{Ax} = \boldsymbol{y}$ может быть сформулирована как нахождение пригодного вектора $\boldsymbol{y}^*$, ближайшего к $\boldsymbol{y}$. Эквивалентной задачей является вычисление векторного расстояния $\boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y})$ между порождаемым матрицей $\boldsymbol{A}$ пространством $\mathrm{Sp}(\boldsymbol{A})$ и вектором $\boldsymbol{y}$. Это расстояние равно сумме $\boldsymbol{y}^* \oplus \boldsymbol{y}$ и может рассматриваться как вектор искажения $\boldsymbol{e}$, если вес последнего (число единиц) меньше чем τ – усредненное кратчайшее расстояние между элементами множества $\mathrm{Sp}(\boldsymbol{A})$. При выполнении этого условия вектор $\boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y})$ определяется обычно однозначно.

Две булевы матрицы называются эквивалентными, если они порождают одно и то же линейное векторное пространство. Заменой произвольного столбца $\boldsymbol{a}^i$ в матрице $\boldsymbol{A}$ его покомпонентной суммой по модулю два с некоторым другим столбцом $\boldsymbol{a}^j$ той же матрицы получается другая матрица $\boldsymbol{A}^+$, эквивалентная исходной матрице $\boldsymbol{A}$. При этом векторное расстояние $\boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}^+), \boldsymbol{y}) = \boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y})$. Аналогично можно заменить вектор $\boldsymbol{y}$ в системе $(\boldsymbol{A}, \boldsymbol{y})$ его суммой с произвольным столбцом $\boldsymbol{a}^j$ матрицы $\boldsymbol{A}$: $\boldsymbol{y}^+ = \boldsymbol{y} \oplus \boldsymbol{a}^j$ $(j = 1, 2, \ldots, n)$, не нарушив равенства $\boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y}^+) = \boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y})$.

Введенные операции могут быть использованы для эквивалентных преобразований системы $(\boldsymbol{A}, \boldsymbol{y})$, не влияющих на решения рассматриваемой задачи, но облегчающих их нахождение. С использованием этих операций можно канонизировать рассматриваемую систему следующим образом.

1. Выбираются n линейно независимых строк в матрице $\boldsymbol{A}$.

2. Путем замены некоторых столбцов матрицы $\boldsymbol{A}$ их покомпонентной суммой по модулю два с другими столбцами добиваются того, чтобы в каждой из выбранных строк было по одной единице (в i-й компоненте для i-й из выбранных строк).

3. В векторе $\boldsymbol{y}$ (путем сложения его с некоторыми определенными столбцами матрицы $\boldsymbol{A}^+$) удаляются единицы во всех компонентах, соответствующих выбранным строкам.

Полученная таким образом система $(\boldsymbol{A}^+, \boldsymbol{y}^+)$ затем сокращается путем удаления всех выбранных строк из матрицы $\boldsymbol{A}^+$ и соответствующих компонент вектора $\boldsymbol{y}^+$. Остающиеся элементы образуют *сокращенную* каноническую форму, представленную булевой $((m - n) \times n)$-матрицей $\boldsymbol{B}$ и $(m - n)$-вектором $\boldsymbol{u}$.

Для упрощения формулировки решаемой задачи введем следующие обозначения:

$\boldsymbol{C}$ – произвольный столбцовый минор матрицы $\boldsymbol{B}$;

$\boldsymbol{c}$ – n-вектор, в котором единицами отмечены входящие в $\boldsymbol{C}$ столбцы из $\boldsymbol{B}$;

$w(\boldsymbol{c})$ – вес булева вектора $\boldsymbol{c}$, т. е. число единиц в нем;

$\oplus(\boldsymbol{C})$ – покомпонентная сумма по модулю два всех столбцов из $\boldsymbol{C}$, равная $\boldsymbol{Bc}$;

$\boldsymbol{s} = \oplus(\boldsymbol{C}) \oplus \boldsymbol{u}$;

$(\boldsymbol{c}, \boldsymbol{s})$ – конкатенация векторов $\boldsymbol{c}$ и $\boldsymbol{s}$.

Задача восстановления системы $(\boldsymbol{A}, \boldsymbol{y})$ с искаженным вектором $\boldsymbol{y}$ сводится к нахождению столбцового минора $\boldsymbol{C}$ матрицы $\boldsymbol{B}$, которому соответствует минимальное значение величины $w(\boldsymbol{c}) + w(\boldsymbol{s})$. В таком случае векторное расстояние $\boldsymbol{d}(\mathrm{Sp}(\boldsymbol{A}), \boldsymbol{y}) = (\boldsymbol{c}, \boldsymbol{s}) = \boldsymbol{e}$ и $\boldsymbol{y}^* = \boldsymbol{y} \oplus \boldsymbol{e}$.

После этого можно восстановить совместную систему $(\boldsymbol{A}, \boldsymbol{y}^*)$ и найти ее единственный корень $\boldsymbol{x}^*$.

2.3. Рандомизированный алгоритм

Основные затраты времени на выполнение описанного выше метода восстановления системы приходятся на поиск столбцового минора $\boldsymbol{C}$ матрицы $\boldsymbol{B}$ с минимальным значением величины $w(\boldsymbol{c}) + w(\boldsymbol{s})$. Эти затраты пропорциональны числу перебираемых подмножеств столбцов матрицы $\boldsymbol{B}$, которое в значительной степени зависит от максимального уровня перебора k, ограничивающего сверху число столбцов в испытываемых подмножествах.

Затраты времени существенно сокращаются при малой степени искажений, когда $w(\boldsymbol{e})$ значительно меньше τ – усредненного кратчайшего расстояния между элементами множества $\mathrm{Sp}(\boldsymbol{A})$. В этом случае решение может *распознаваться* по весу на некотором более низком уровне перебора k.

Дополнительный выигрыш можно получить путем декомпозиции процесса решения, при которой вместо одной канонической формы матрицы $\boldsymbol{A}$ рассматриваются p форм, порожденных булевыми векторами $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_p$, задающими различные случайно выбираемые максимальные линейно независимые подмножества строк в матрице $\boldsymbol{A}$. Таким образом, процесс решения *рандомизируется* [7]. Решение ищется «параллельно» по всем формам, сначала на уровне перебора 0, затем на уровне 1 и т. д., пока на очередном уровне k по какой-нибудь форме не будет найдено решение с весом w, удовлетворяющим условию $w < \tau - 1$.

Шансы на обнаружение решения с весом w на пониженном уровне перебора k оцениваются следующим образом. Существуют C_m^w различных значений булева m-вектора $\boldsymbol{e}$ с w единицами. Предполагается, что все эти значения равновероятны. Тогда число тех из них, которые имеют ровно k единиц ($k \le n < m$) в n-векторе $\boldsymbol{c}$, выделенном в m-векторе $\boldsymbol{e}$, оценивается формулой $\mathrm{C}_n^k \mathrm{C}_{m-n}^{w-k}$. Иными словами, это число составляет $100\ \mathrm{C}_n^k \mathrm{C}_{m-n}^{w-k} / \mathrm{C}_m^w$ % от числа всех значений вектора $\boldsymbol{e}$ с w единицами.

Очевидно, что выбор формы $\boldsymbol{f}_i$, которой соответствует минимальное значение величины $w(\boldsymbol{c}_i)$, позволяет значительно снизить уровень перебора и тем самым существенно ускорить процесс поиска решения.

Для дальнейших исследований рандомизированного алгоритма, изложенного в статье [7], полагается, что работа его заканчивается поиском кратчайшего m-вектора искажений $\boldsymbol{e}$, а внутренний трудоемкий цикл алгоритма, инвариантный по отношению к различным интерпретациям цели, которая преследуется при решении переопределенной СЛЛУ, выделяется особо и называется далее алгоритмом R. Исходными данными для него (в обозначениях, принятых в разд. 2.2) служат массив из $((m-n) \times n)$-матриц $\boldsymbol{B}_i$, массив из $(m-n)$-векторов $\boldsymbol{u}_i$ ($i = 1, 2, \ldots, p$) и оценка $\tau(m, n)$ верхней границы веса решения $w(\boldsymbol{e})$. Результатом работы алгоритма R являются n-вектор $\boldsymbol{c}$, а также номер f сокращенной канонической формы и уровень перебора k, на которых этот вектор был найден.

Напомним, что вектор $\boldsymbol{c}$ своими k единицами отмечает столбцовый минор матрицы $\boldsymbol{B}_f$, обеспечивающий минимальное значение веса $w(\boldsymbol{e})$ искомого m-вектора искажений $\boldsymbol{e}$. Вектор $\boldsymbol{e}$ однозначно определяется по f-й канонической форме, восстановленной из сокращенной по m-вектору $\boldsymbol{f}_f$, который представляет подмножество независимых строк исходной матрицы $\boldsymbol{A}$, образующих f-ю каноническую форму.

Объем вычислений при нахождении кратчайшего n-вектора $\boldsymbol{c}$ на k-м уровне f-й канонической формы пропорционален числу $M(n, p, f, k)$ рассматриваемых подмножеств столбцов в p канонических формах:

$$M(n, p, f, k) = p \sum_{i=0}^{k-1} \mathrm{C}_n^i + (f - 0{,}5)\, \mathrm{C}_n^k . \qquad (1)$$

При этом предполагается, что на f-й форме рассматривается половина из C_n^k подмножеств с мощностью k.

3. Параллельная реализация рандомизированного алгоритма

Параллелизм, заложенный в алгоритме *R*, легко реализуется на мультипроцессорной системе семейства СКИФ [8]. Необходимо лишь обеспечить среду, для которой он был разработан; сформировать заданное число эквивалентных канонических форм, распределив их между процессами, участвующими в параллельном решении; дождаться, когда какой-нибудь из них (самый «быстрый») найдет решение, оформить его надлежащим образом и заставить все остальные процессы прекратить свою работу.

Для проведения серийных испытаний рандомизированного алгоритма на потоке псевдослучайных переопределенных СЛЛУ $\boldsymbol{Ax} = \boldsymbol{y}$ с заданными параметрами разработан специальный полигон, обеспечивающий прием значений параметров, генерирование псевдослучайных СЛЛУ с заданными параметрами, их решение и оформление итогов решения в надлежащем виде. При этом используются следующие параметры:

m – число уравнений;

n – число переменных;

d – процент искаженных компонент в векторе правых частей $\boldsymbol{y}$;

p – число канонических форм;

z – «зерно» генератора для первой канонической формы;

g – «зерно» генератора для системы в целом;

v – номер варьируемого в серии параметра ($v \in \{1, 2, \ldots, 6\}$);

l – число шагов в серии;

h – величина шага для изменения значений варьируемого параметра.

В серийных испытаниях принимают участие $N + 1$ процессов. Один из них, называемый *корневым*, управляет испытаниями и при необходимости, если для сравнения задан режим однопроцессорного исполнения алгоритма *R*, реализует его, применяя ко всем p сокращенным каноническим формам. Остальные N процессов, называемые *подчиненными*, реализуют параллельную работу алгоритма *R*, обеспечивая поиск m-вектора искажений $\boldsymbol{e}$, каждый на своем подмножестве канонических форм.

В системах с индивидуальной памятью, к каким относится семейство СКИФ, информационное взаимодействие между процессами осуществляется через обмен сообщениями. При организации обмена сообщениями между процессами используется специально разработанный для этих целей класс **CBUF** операций над сообщениями, обеспечивающий образование приемопередающих буферов для сообщений заданной емкости, а также компактную упаковку логических объектов в посылаемое сообщение и распаковку их из принятого сообщения [9].

Далее приводится перечень операций, выполняемых корневым и подчиненными процессами во время серийных испытаний рандомизированного алгоритма, параллельная реализация которого была разработана в среде MPI [10, 11].

Корневой процесс заключается в следующем:

1. У испытателя запрашивается режим работы и значения параметров потока псевдослучайных переопределенных СЛЛУ, решаемых в данной серии.

2. Генерируется очередная $m \times n$ матрица $\boldsymbol{A}$, и для нее формируется транспонированная $n \times m$ матрица $\boldsymbol{D} = \boldsymbol{A}^{\mathrm{T}}$. Генерируется n-вектор $\boldsymbol{x}$ и по нему получается (с использованием матрицы $\boldsymbol{D}$) пригодный m-вектор $\boldsymbol{y}^* = \boldsymbol{Ax}$. Генерируется m-вектор искажений $\boldsymbol{e}'$ с $m\,d\,/\,100$ единицами, и по нему из вектора $\boldsymbol{y}^*$ получается искаженный m-вектор правых частей $\boldsymbol{y} = \boldsymbol{y}^* \oplus \boldsymbol{e}'$.

3. Если указан режим параллельного исполнения алгоритма *R*, всем N подчиненным процессам посылается сообщение, содержащее три числовых параметра (p – число канонических форм, z – «зерно» генератора для первой канонической формы и значение $\tau(m, n)$ верхней границы веса решения $w(\boldsymbol{e})$), а также $n \times m$ матрицу $\boldsymbol{D}$ и искаженный m-вектор правых частей $\boldsymbol{y}$.

4. Если для сравнения задан режим однопроцессорного исполнения алгоритма *R*, то он реализуется в применении ко всем p сокращенным формам. Фиксируются результаты его работы

(номер формы f и уровень перебора k, на которых находится n-вектор $\boldsymbol{c}$) и значение m-вектора искажений $\boldsymbol{e}$, вычисляется его вес $w(\boldsymbol{e})$ и отмечается общее время получения t_s.

5. В случае параллельного исполнения алгоритма R переходит в состояние ожидания ответных сообщений от подчиненных процессов (п. 6), а иначе переходит к п. 8.

6. Получив сообщение от одного из подчиненных процессов, быстрее всех нашедшего решение, процесс оформляет его надлежащим образом, фиксирует его параметры, в том числе время решения t_p, посылает всем подчиненным процессам сигнал конца работы (END) и переходит в состояние ожидания и приема еще не принятых сообщений от других процессов (п. 7).

7. Принимаются еще не принятые от подчиненных процессов сообщения. Переход к п. 8.

8. Если решенная СЛЛУ является последней в серии испытуемых, то работа заканчивается, а иначе вычисляется новое значение варьируемого параметра, оставляя значения остальных прежними, и осуществляется возврат к п. 2.

При ***подчиненном процессе*** ожидается сообщение от корневого процесса. Если в полученном сообщении оказался сигнал END, то работа заканчивается, а иначе на основании полученной информации (параметров p, z, τ, матрицы $\boldsymbol{D}$ и вектора $\boldsymbol{y}$) конструируются b канонических форм, к ним применяется параллельный вариант алгоритма R и корневому процессу посылается ответное сообщение с результатами решения. При этом $b = [p/N] + 1$ для первых v процессов и $b = [p/N]$ для остальных $N - v$ процессов, где $[p/N]$ – целая часть от деления p на N, а v – остаток от этого деления. Ответное сообщение содержит m-вектор искажений $\boldsymbol{e}$, номер формы f и уровень перебора k, на которых был найден n-вектор $\boldsymbol{c}$, а также время t_p, затраченное на поиск и оформление решения. Если во время своей работы алгоритм R обнаруживает, что корневой процесс послал сигнал END, то прерывает работу с выдачей значения $k = -1$.

Заметим, что последовательный и параллельный варианты алгоритма R отличаются лишь тем, что во внутренний цикл параллельного варианта, который выполняется в подчиненных процессах, вставлена операция *MPI_IProbe* [10, 11], реагирующая на факт посылки корневым процессом сигнала END. В случае «положительной пробы» (когда к моменту проверки корневой процесс уже послал сигнал END) подчиненный процесс должен обеспечить прием этого сигнала и прекратить работу. Место вставки операции *MPI_IProbe* выбрано с таким расчетом, чтобы обеспечить оптимальную скорость реакции подчиненных процессов на сигнал END при допустимых накладных расходах.

4. Экспериментальные испытания

Рандомизированный алгоритм R, в последовательной и параллельной его модификациях, был подвергнут обширным экспериментальным испытаниям на потоке псевдослучайных СЛЛУ с использованием описанного в разд. 3 испытательного полигона. Основная масса испытаний была проведена на модели полигона, реализованного на персональном компьютере, а затем наиболее представительные эксперименты были повторены на мультипроцессорной системе семейства СКИФ. Как и следовало ожидать, наибольшее влияние на время решения СЛЛУ оказывают процент d искаженных компонент в векторе правых частей $\boldsymbol{y}$, число p формируемых канонических форм, а также число N подчиненных процессов, участвующих в решении.

Результаты некоторых экспериментов, выполненных на мультипроцессорной системе семейства СКИФ, представлены в табл. 1–6, где через t_s и t_p обозначены временные затраты (с) на решение СЛЛУ последовательным и параллельным вариантами алгоритма R соответственно. Кроме времени решения СЛЛУ в таблицах приводятся также номера канонических форм f и уровни перебора k, на которых были найдены кратчайшие решения с весом $w(\boldsymbol{e}) = m\,d\,/100$.

В табл. 1–3 приведены результаты решения псевдослучайных СЛЛУ при изменении процента d искаженных компонент в векторе $\boldsymbol{y}$ и числа N подчиненных процессов, участвующих в решении, с фиксированием значений остальных параметров.

Таблица 1

Результаты решения СЛЛУ с различными значениями процента d искаженных компонент в векторе $\boldsymbol{y}$ и различными числами N подчиненных процессов, участвующих в решении (m = 800, n = 70, p = 20, z = 1, g = 5)

d	f	k	t_s	N = 5		N = 10		N = 15		N = 20	
				t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p
10	6	3	3,85	0,82	4,70	0,66	5,83	0,66	5,83	0,14	27,50
12	15	4	125,27	20,14	**6,22**	2,69	**46,57**	2,16	**57,99**	2,16	**57,99**
14	15	5	1687,81	271,55	**6,22**	39,23	**40,02**	30,96	**54,52**	31,03	**54,39**

Таблица 2

Результаты решения СЛЛУ с различными значениями процента d искаженных компонент в векторе $\boldsymbol{y}$ и различными числами N подчиненных процессов, участвующих в решении (m = 800, n = 80, p = 40, z = 1, g = 14)

d	f	k	t_s	N = 5		N = 10		N = 15		N = 20	
				t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p
10	28	2	3,21	0,53	**6,00**	0,32	10,03	0,18	**17,83**	0,16	20,06
12	37	4	509,44	67,20	**7,58**	11,93	**42,70**	10,43	**48,84**	10,43	**48,84**
14	37	5	8083,57	985,07	**8,21**	133,89	**60,37**	106,64	**75,80**	106,44	**75,94**

Таблица 3

Результаты решения СЛЛУ с различными значениями процента d искаженных компонент в векторе $\boldsymbol{y}$ и различными числами N подчиненных процессов, участвующих в решении (m = 1000, n = 80, p = 40, z = 1, g = 1)

d	f	k	t_s	N = 5		N = 10		N = 15		N = 20	
				t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p
10	5	3	7,77	4,22	1,84	0,49	15,86	1,22	6,37	0,29	26,79
11	5	4	104,08	26,61	3,91	5,71	18,23	20,82	5,00	3,79	27,46
12	34	4	572,48	26,62	**21,51**	22,87	**25,03**	21,02	**27,24**	21,17	**27,04**
13	34	5	8855,79	439,96	**20,13**	370,78	**23,88**	336,32	**26,33**	337,22	**26,26**

С возрастанием процента d пропорционально увеличивается вес $w(\boldsymbol{e})$ решения СЛЛУ, при этом уменьшается вероятность нахождения его на низком уровне перебора k, что в конечном итоге и определяет быстрый рост временных затрат. С увеличением числа N подчиненных процессов уменьшается число канонических форм, среди которых каждый из них осуществляет поиск решения, что приводит к сокращению времени t_p и к увеличению отношения t_s/t_p. При малом времени счета (до 10 с) значительная доля его приходится на организацию межпроцессорных информационных обменов, которая быстро уменьшается с возрастанием суммарного времени счета. При этом значительно увеличивается и отношение t_s/t_p.

Следует заметить, что коэффициент $K = t_s/t_p$ выигрыша по быстродействию параллельного варианта исполнения алгоритма R перед последовательным зависит от того, на какое место φ ($0 < \varphi \le p/N$) при распределении p канонических форм по N подчиненным процессам попадет f-я форма с кратчайшим решением. Воспользовавшись приведенной в разд. 2.3 формулой (1) для оценки величин t_s и t_p, легко показать, что при $N(2\varphi - 1) < (2f - 1)$

$$K = t_s/t_p > N. \qquad (2)$$

В табл. 4 представлены значения величин, участвующих в проверке условия (2) для параметров из табл. 3. Значения $N(2\varphi - 1)$, удовлетворяющие условию (2), и соответствующие им значения $K = t_s/t_p$ отмечены в таблицах жирным шрифтом.

В табл. 5–6 приведены результаты решения псевдослучайных СЛЛУ при изменении числа p канонических форм и числа N подчиненных процессов, участвующих в решении, с фиксированием значений остальных параметров. С возрастанием числа p увеличивается вероятность найти решение на более низком уровне перебора k, что и способствует снижению временных затрат.

Таблица 4

Результаты проверки условия $N(2\varphi - 1) < (2f - 1)$ для табл. 3 (при $m = 800$, $n = 80$, $p = 40$, $z = 1$, $g = 14$)

f	$2f$–1	$N = 5$ $p = 8 \times 5$			$N = 10$ $p = 4 \times 10$			$N = 15$ $p = 3 \times 10 + 2 \times 5$			$N = 20$ $p = 2 \times 20$		
		φ	2φ–1	$N(2\varphi$–1)	φ	2φ–1	$N(2\varphi$–1)	φ	2φ–1	$N(2\varphi$–1)	φ	2φ–1	$N(2\varphi$–1)
5	9	5	9	45	1	1	10	2	3	45	1	1	20
5	9	5	9	45	1	1	10	2	3	45	1	1	20
34	67	2	3	**15**	2	3	**30**	2	3	**45**	2	3	**60**
34	67	2	3	**15**	2	3	**30**	2	3	**45**	2	3	**60**

Таблица 5

Влияние числа p канонических форм и числа N подчиненных процессов на работу алгоритма R при $m = 800$, $n = 80$, $d = 12$, $z = 1$, $g = 19$

p	f	k	t_s	$N = 5$		$N = 10$		$N = 15$		$N = 20$	
				t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p
40	33	5	7052,28	218,64	**32,26**	155,75	**45,28**	128,99	**54,67**	128,29	**54,97**
60	59	4	812,08	149,69	**5,43**	66,82	**12,15**	39,17	**20,73**	25,47	**31,88**
80	79	3	60,80	10,96	**5,55**	4,87	**12,48**	2,65	**22,94**	1,86	**32,69**

Таблица 6

Влияние числа p канонических форм и числа N подчиненных процессов на работу алгоритма R при $m = 1000$, $n = 80$, $d = 12$, $z = 1$, $g = 10$

p	f	k	t_s	$N = 5$		$N = 10$		$N = 15$		$N = 20$	
				t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p	t_p	t_s/t_p
40	40	5	10541,93	1922,72	**5,48**	870,34	**12,11**	345,07	**30,55**	345,14	**30,54**
60	51	4	870,40	51,26	**16,98**	45,71	**19,04**	43,78	**19,88**	42,90	**20,29**
80	78	3	78,08	12,81	**6,10**	5,31	**14,70**	2,49	**31,36**	1,56	**50,05**

Заключение

Проведенные эксперименты с рандомизированным алгоритмом R показывают, что использование мультипроцессорных систем может существенно ускорить процесс решения несовместных систем линейных логических уравнений. При решении больших СЛЛУ на мультипроцессорной системе семейства СКИФ параллельная модификация алгоритма R с N подчиненными процессорами может выиграть по быстродействию у последовательной модификации на одном процессоре более чем в N раз.

Список литературы

1. Закревский А.Д. Комбинаторные методы оптимизации решений систем линейных логических уравнений // Вестник ТГУ. Приложение. № 6. – Сентябрь 2003. – С. 4–8.

2. Gauss C. F. Beitrage zur Theorie der algebraischen Gleichungen. – Germany, 1849.

3. Закревский А.Д., Торопов Н.Р. Полиномиальная реализация частичных булевых функций и систем. – М.: УРСС, 2003.

4. Zakrevskij A. Solution of a system of logical equations with distorted right members – when it could be found. – New Information Technologies // Proc. of the Fifth International Conference NITe'2002. – Minsk: BSEU, 2002. – Vol. 1. – P. 54–58.

5. Закревский А.Д., Торопов Н.Р. Поиск кратчайшего решения системы линейных логических уравнений // Тр. Второй Междунар. конф. CAD DD'1997. Т. 2. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1997. – С. 16–23.

6. Закревский А.Д. Эффективные методы нахождения кратчайших решений систем линейных логических уравнений // Проблемы управления. – 2003. – № 4. – С.16 – 22.

7. Zakrevskij A.D. Randomization of a parallel algorithm for solving undefined systems of linear logical equations // Proc. of the International Workshop on Discrete-Event System Design – DESDes'04. – University of Zielona Gora Press, Poland, 2004. – P. 97–102.

8. Принципы построения суперкомпьютеров семейства «СКИФ» и их реализация / С.М. Абрамов, Н.Н. Парамонов, В.В. Анищенко, С.В. Абламейко // Информатика. – 2004. – № 1. – С. 89–106.

9. Торопов Н.Р. Параллельные логико-комбинаторные вычисления в среде MPI // Информатика. – 2005. – № 3. – С. 82–90.

10. Gropp W. , Lusk E. , Skjellum A. Using MPI: Portable Parallel Programming with the Message Passing Interface. – MIT Press, 1995.

11. Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. – Мн.: БГУ, 2002. – 323 с.

Поступила 24.10.05

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: zakr@newman.bas-net.by

A.D. Zakrevskij, N.R. Toropov

SOLVING THE MAJORITY OF LINEAR LOGICAL EQUATIONS OF INCONSISTENT SYSTEM ON SUPERCOMPUTER

Overdefined systems of linear logical equations are considered, the number of equations in which surpasses the number of variables. As a rule, such a system has no root and therefore is called inconsistent. Anyhow, these systems can be solved in some sense. For example, from the cryptography point of view, it is interesting to find solutions satisfying the maximum number of equations or, if the right parts are distorted, to restore the system. The parallel implementation of a randomized algorithm is suggested for solving inconsistent systems on the supercomputer SKIF. The results of experiments testify the efficiency of parallel computations when solving large systems of linear logical equations.